UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Implementación de una herramienta orientada a niños para el aprendizaje del desarrollo de videojuegos 3D

Guillermo Álvarez Ocaña
Tutor: Carlos Aguirre Maeso

Junio 2021

Implementación de una herramienta orientada a niños para el aprendizaje del desarrollo de videojuegos 3D

AUTOR: Guillermo Álvarez Ocaña
TUTOR: Carlos Aguirre Maeso

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2021

Resumen (castellano)

En este Trabajo de Fin de Grado se desarrolla un editor 3D para la creación de escenas de videojuegos, generando una herramienta que funciona en el navegador web. El editor está orientado a niños, sin conocimientos previos de informática, para introducirles en el desarrollo de videojuegos 3D.

Mediante este editor, los usuarios, podrán empezar a interactuar con este tipo de entornos de desarrollo y familiarizarse con algunos conceptos básicos en la creación de una escena 3D.

El editor 3D se puede ejecutar en cualquier navegador web actual mediante un servidor local y ha sido desarrollado mediante los lenguajes: JS (JavaScript), HTML5 (*HyperText Markup Language*) y CSS3 (*Cascading Style Sheets*).

La aplicación constará de una GUI (*Graphical User Interface*) sencilla, sin demasiada información en pantalla, fácil e intuitiva de usar. Ya que el objetivo principal es lograr una aplicación orientada a niños.

El editor 3D permite al usuario realizar:

- La creación de una escena 3D mediante el uso de *assets* ofrecidos en el propio editor.
- La traslación, rotación y cambio de escala de los objetos 3D de la escena.
- La inserción de los *assets* en la escena 3D mediante el uso del método Drag & Drop.
- La eliminación de elementos de la escena 3D.
- El movimiento libre de la cámara mediante uso del ratón.

El proyecto realizado, es por tanto, un entorno de desarrollo ejecutado en la web, con una orientación enfocada a niños sin conocimientos previos de informática. Por ello el editor 3D desarrollado ofrece a los usuarios la funcionalidad necesaria para que este editor 3D sirva como aprendizaje básico al desarrollo de videojuegos 3D.

Palabras clave (castellano)

Editor 3D, JavaScript, HTML, CSS, GUI

Abstract (English)

In this Final Degree Project a 3D editor is developed to create video game scenes, generating a tool that works in the web browser. The editor is aimed at children, without previous knowledge in computer science, to introduce them to the development of 3D video games.

Using this editor, they can have a first and straight forward contact with a game development environment and become familiar with some basic concepts in creating a 3D scene.

The 3D editor can be run in any web browser through a local server and has been developed using the languages of JS (JavaScript), HTML5 (HyperText Markup Language) and CSS3 (Cascading Style Sheets).

The application will consist of a simple GUI (Graphical User Interface), with little information on the screen making it easy and intuitive to use.

The 3D editor allows the user to:

- Create a 3D scene using assets offered in the editor itself.
- Perform the translation, rotation and scalability of the 3D objects in the scene.
- Use the Drag & Drop method to introduce the assets in the 3D editor scene.
- Eliminate elements from the 3D scene.
- And to freely move the camera using the mouse.

The project carried out, is therefore, a game development environment executed on the web, with a novel population focus: children without previous knowledge in computing sciences. More particularly, the 3D editor offers a forthright learning tool with the necessary functions in order to develop 3D videogames.

Keywords (english)

Editor 3D, JavaScript, HTML, CSS, GUI

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Los videojuegos ya no son un mero producto de entretenimiento para un pequeño público de consumidores los cuales en su día podían permitirse tener un ordenador o una consola y usar una pequeña pausa de su tiempo libre para jugar un rato a un videojuego. Hoy en día el mercado de los videojuegos se ha convertido en una gran potencia, ofreciendo grandes ingresos a múltiples empresas. Incluso se ha creado una nueva profesión de jugadores de videojuegos, gracias a la creación de equipos profesionales de videojuegos respaldados por el patrocinio de empresas y por beneficios obtenidos en competiciones con el premio ofrecido por las propias empresas desarrolladoras del juego de dicha competición.

Es por esto que hoy en día el desarrollo de videojuegos está en auge y son muchas las empresas que intentan hacerse un hueco en este mercado, o incluso pequeños grupos de programadores que intentan desarrollar un juego con la esperanza de ser el próximo juego exitoso.

Existe una gran variedad de distintos entornos de desarrollo para la realización de videojuegos 3D, así como herramientas para la creación de escenarios 3D para los mismos juegos o para otras tareas ajenas al desarrollo de los videojuegos. Podemos encontrar un gran abanico de opciones, desde herramientas desarrolladas por equipos de programadores profesionales, hasta herramientas amateur. Pese a las múltiples opciones presentes de entornos de desarrollo 3D, la mayoría de ellas requieren de unos conocimientos previos de lenguajes de programación que dicho entorno de desarrollo soporte o incluso conocimientos únicos para poder emplear dicho entorno de desarrollo.

Por ello hemos desarrollado una herramienta sencilla para navegador orientada a niños, que permita la creación de una escena 3D simple sin la necesidad de ningún conocimiento previo en lenguajes de programación y con la idea de que esta herramienta sirva como una introducción al desarrollo de videojuegos 3D.

1.2 Objetivos

Este proyecto tiene como objetivo desarrollar una herramienta de manejo simple y orientada a niños para la creación de escenas 3D en el navegador. Este editor 3D ha sido desarrollado con HTML5, CSS3 y JavaScript, haciendo uso específico de la librería Three.js [1].

La idea principal del proyecto es realizar un editor 3D que no necesite de ningún conocimiento previo de informática para su uso y que sirva para instruir a niños sobre ciertos conceptos en la creación de una escena 3D con la ayuda de un limitado número de objetos 3D para crear el escenario y familiarizarse con conceptos como la traslación, rotación y cambio de escala de objetos 3D en la escena.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:** este capítulo contiene la motivación, los objetivos del trabajo realizado y la organización de la memoria.
- **Estado del arte:** en este capítulo se realiza un estudio sobre diferentes aplicaciones existentes que presentan una relación con este trabajo.
- **Diseño:** en este capítulo se define el análisis previo y el diseño general de la aplicación.
- **Desarrollo:** este capítulo explica como ha sido desarrollada la aplicación tratando exclusivamente el desarrollo de código.
- **Desarrollo visual:** este capítulo explica como ha sido desarrollada la aplicación tratando únicamente el apartado visual.
- **Integración, pruebas y resultados:** en este capítulo se explica como se han realizado las pruebas de la aplicación según se progresaba en el desarrollo de la aplicación y los resultados obtenidos.
- **Conclusiones y trabajo futuro:** en este capítulo se tratará sobre las conclusiones finales del proyecto y futuras mejoras que recibirá la aplicación.

2 Estado del arte

Como ya se ha mencionado con anterioridad en este documento, son muchas las opciones existentes en cuanto a entornos de desarrollo de videojuegos 3D se refiere, y no solo eso, hoy en día incluso podemos encontrar videojuegos que en sí mismos sirven como entornos para crear nuevos videojuegos con las capacidades limitadas ofrecidas por dicho videojuego. Algunos ejemplos de este tipo pueden ser juegos como Roblox [2] o el modo creativo de Fornite [3], aunque existen más videojuegos de este tipo que están surgiendo actualmente en el mercado.

Para la realización de nuestro propio entorno de desarrollo hemos realizado un análisis de los distintos entornos de desarrollo de videojuegos 3D más usados a día de hoy, ya que lo que nos interesa en este apartado es realizar un análisis de las distintas GUI (*Graphical User Interface*) para finalmente desarrollar nuestra propia GUI para nuestro entorno de desarrollo. Dado que nuestro proyecto está enfocado a realizar una herramienta de fácil manejo y orientada a niños, no solo analizaremos los entornos más utilizados, sino que también analizaremos algunos entornos más básicos que puedan servirnos de inspiración para la realización de nuestra herramienta.

2.1 Entornos populares

En este apartado se realizará un análisis de los tres entornos profesionales más populares de desarrollo de videojuegos 3D, Unity [4], CryEngine [5] y Unreal Engine [6].

Analizando los editores 3D de estos tres entornos de desarrollo se puede obtener una idea de como organizan la pantalla del editor buscando similitudes entre los mismos y enfrentando estos entornos sobre nuestros objetivos del editor. Una vez hecho esto, se podrían sacar conclusiones sobre que elementos serían óptimos realizar en nuestro propio editor 3D y que elementos no son convenientes pese a que estos editores posean dichos elementos o distribución de los mismos.

2.1.1 Unity

Unity [4] es uno de los entornos de desarrollo más utilizados, no solo por empresas profesionales en el mercado de los videojuegos, sino también por pequeños grupos de programadores amateur o incluso para proyectos desarrollados por una única persona. Es un entorno de desarrollo bastante completo ya que posee mucha funcionalidad básica estandarizada para la creación de videojuegos e incluso tiene una tienda interna de *assets*.

La herramienta ofrece la utilización de *scripts* en C# como lenguaje de programación principal. Por otro lado, Unity [4] ofrece la opción para que el producto que se realiza sea implementado para múltiples plataformas.

El editor se divide en distintas secciones como se puede ver en la Figura 2.1. El usuario tiene cierta libertad para colocar dichas secciones en la parte de la pantalla que más le guste. Las secciones más importantes son las que poseen la escena y la sección con la cámara del juego, sección muy importante para poder ver como todo el desarrollo va tomando forma en ejecución. Por otro lado, alrededor de la escena principal tenemos distintas secciones en las que nos encontramos una gran cantidad de información sobre nuestro proyecto. Algunas de las secciones que podemos tener a priori son: los objetos que forman dicha escena, las propiedades del objeto seleccionado en cuestión, una consola y un árbol jerárquico con las carpetas del proyecto.

Figura 2.1: Ventana Editor Unity

Podemos observar que Unity [4] muestra un alto contenido de información. La información que ofrece Unity [4] está distribuida en múltiples secciones, y esto puede dar lugar a que al principio sea un problema el familiarizarse con la GUI de este entorno de desarrollo. Por lo que cabe destacar que no es una herramienta sencilla de utilizar sin ciertos conocimientos previos.

Las escenas se crean con el añadido de objetos 3D y *assets* a través de la barra de menú que se encuentra arriba y a la izquierda (común en todas las herramientas). Se pueden ir añadiendo elementos a la escena y posteriormente añadir funcionalidad específica a cada elemento a través de *scripts*.

Unity [4] tiene una característica bastante novedosa en la herramienta llamada Bolt Visual Scripting, se puede ver un ejemplo en la Figura 2.2. Esta característica se puede añadir como *asset* a nuestro proyecto y tener la capacidad de realizar visual scripting de una forma parecida a los Blueprint de Unreal Engine [6] pero para lógica de bajo nivel, por lo que se suele utilizar únicamente para prototipos y verificar el correcto funcionamiento de los elementos en la escena. Por lo que si se desea realizar una funcionalidad detallada y más compleja es necesario para el usuario conocer y realizar *scripts* en C#.

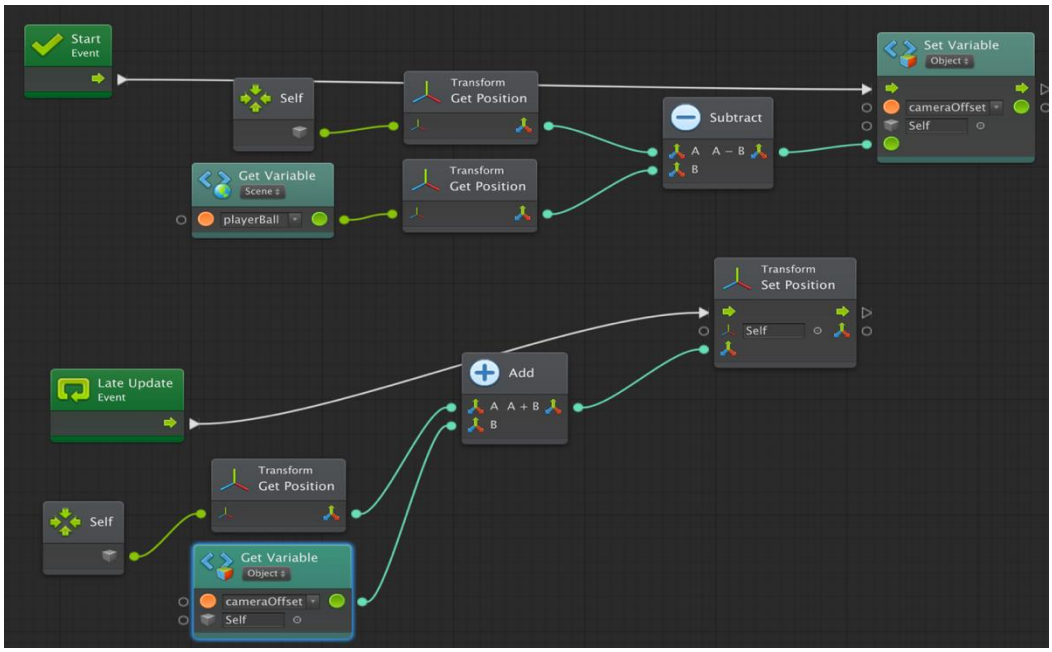

Figura 2.2: Bolt Visual Scripting. [7]

2.1.2 CryEngine

CryEngine [5] es un gran editor 3D con una gran potencia, desarrollado por Crytek, y ha ganado mucha fama con los años gracias a los videojuegos de dicha empresa, usando esta herramienta como entorno de desarrollo de los videojuegos y su propio motor gráfico. Visualmente el editor CryEngine [5] no difiere mucho del resto de editores, como se puede ver en la Figura 2.3. El editor se compone por distintas secciones, con una sección principal formada por la escena del editor, y las otras secciones contienen información de los elementos de la escena y del propio proyecto. Las secciones se pueden organizar al gusto del usuario, modificando su posición y tamaño en la ventana.

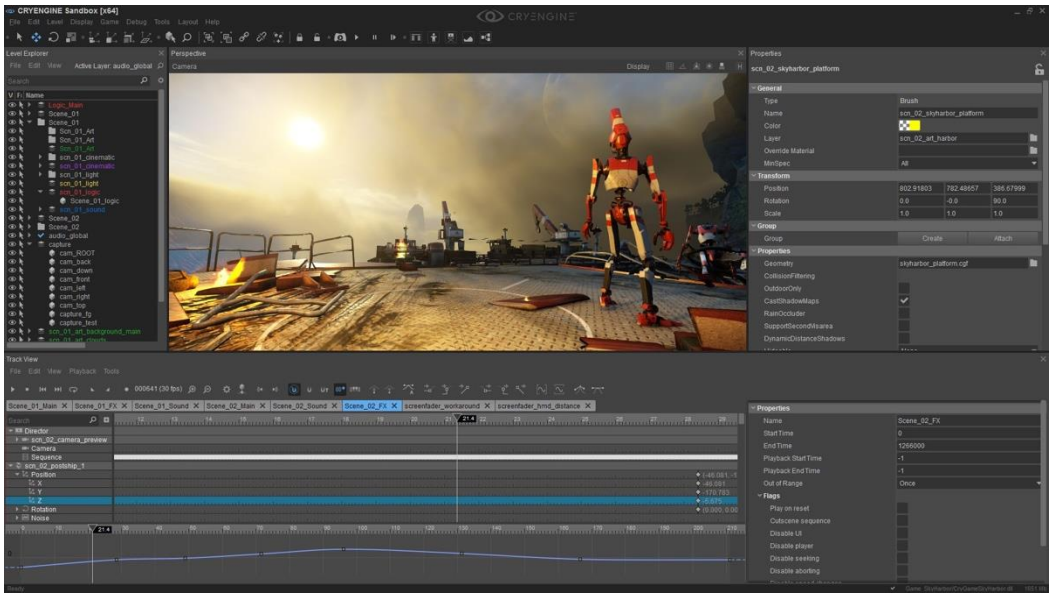

Figura 2.3: Ventana Editor CryEngine. [8]

CryEngine [5] permite añadir funcionalidad a base de *scripts* desarrollados en C++ o C#, y también posee un método de *visual scripting* denominado Flow Graph, es posible ver un ejemplo en la Figura 2.4. Flow Graph puede ayudar a realizar cierta funcionalidad a usuarios sin las capacidades necesarias para realizar *scripts* en C++ o C#.

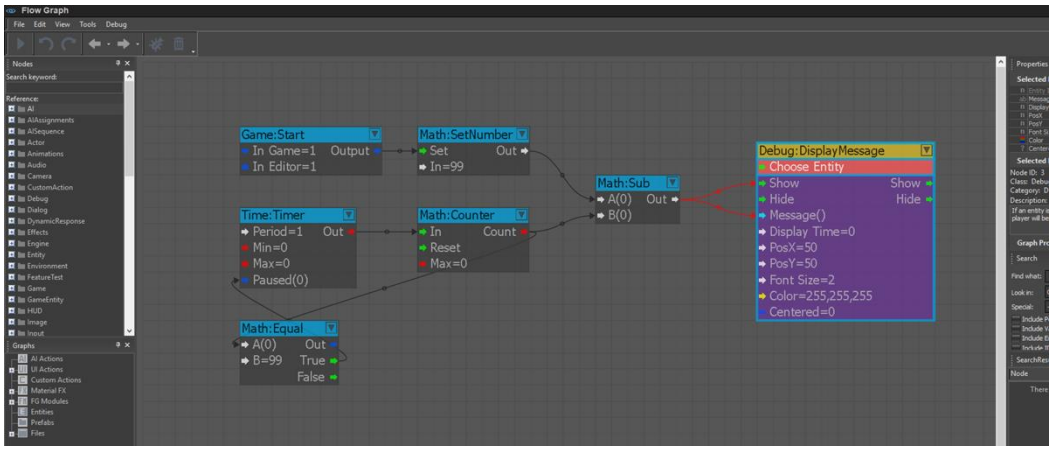

Figura 2.4: Flow Graph CryEngine. [9]

2.1.3 Unreal Engine 4

Unreal Engine [6], desarrollado por Epic Games, es actualmente uno de los entornos de desarrollo más populares. Unreal Engine [6] tiene como lenguaje de programación principal C++ y tiene soporte para C++ y Blueprints en el apartado de *scripting*.

El diseño de la herramienta es muy similar a los entornos de desarrollo descritos con anterioridad, es posible ver un ejemplo en la Figura 2.5. Consta de una sección principal central la cual contiene la escena 3D, y otras secciones más pequeñas que contienen distintas propiedades de los elementos de la escena. Al igual que en Unity [4] y que en CryEngine [5], el usuario tiene cierta libertad para organizar las distintas secciones en la posición que desee y con el tamaño que el usuario estime conveniente.

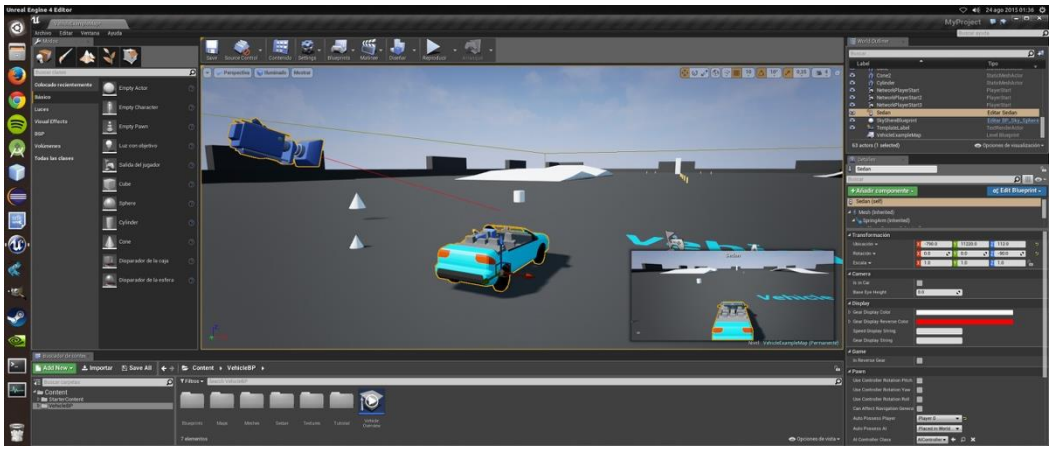

Figura 2.5: Ventana Editor Unreal Engine. [10]

Al igual que Unity [4], Unreal Engine [6] tiene una tienda digital interna que permite a los desarrolladores obtener y proveer ciertos elementos como *assets* y sonidos entre otros, los cuales pueden ser comprados por otros desarrolladores para su uso.

A diferencia de los otros dos editores antes expuestos, Unreal Engine [6] es algo más visual, al mostrar muchos más iconos sobre el tipo de figura que se quiere añadir a la escena. Esto facilita en buena medida la búsqueda de elementos.

Además de la forma clásica de añadir nueva funcionalidad al desarrollo mediante *scripts*, en este caso en el lenguaje C++, Unreal Engine [6] tiene un método visual de *scripting* llamado Blueprints, es posible ver un ejemplo en la Figura 2.6. Este método es una forma de realizar la funcionalidad desarrollada mediante *scripts* en C++, pero esta vez realizando dicha funcionalidad visualmente, esto puede servir de ayuda para usuarios que no tengan los conocimientos de programación necesarios para realizar dicha funcionalidad mediante métodos de *scripting*.

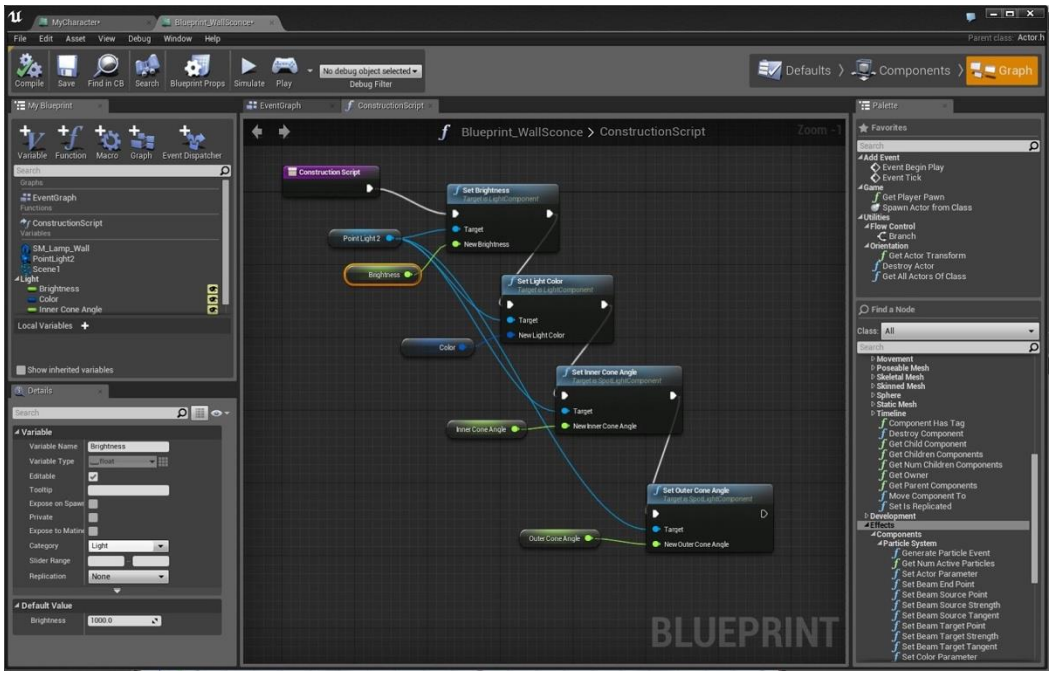

Figura 2.6: Blueprint Unreal Engine. [11]

2.2 Otros entornos a destacar

En este apartado se analizarán dos entornos a destacar, ya que son muy afines a este trabajo, el editor de Three.js [12] y Struckd 3D Game Creator [13].

2.2.1 Three.js

El editor de Three.js [12] es menos complejo que los anteriores, pero se le ha otorgado una gran importancia al análisis de este editor, ya que está creado con la misma librería de JavaScript que el editor 3D realizado en este proyecto.

El editor de Three.js [12] está desarrollado por los autores de la librería de Three.js [1], funciona plenamente en el navegador y utiliza principalmente WebGL para la renderización de los gráficos 3D.

La ventana de la herramienta es más sencilla que las anteriormente analizadas, es posible ver la ventana en la Figura 2.7. Tiene una gran sección central, la cual contiene la escena 3D con un apartado lateral de iconos para seleccionar si se quiere trasladar, rotar o modificar la escala del objeto seleccionado. En la sección situada a la derecha de la ventana, se observan tres apartados distintos: escena, proyecto y configuración, los cuales se solapan entre ellos para no ocupar más espacio en la ventana.

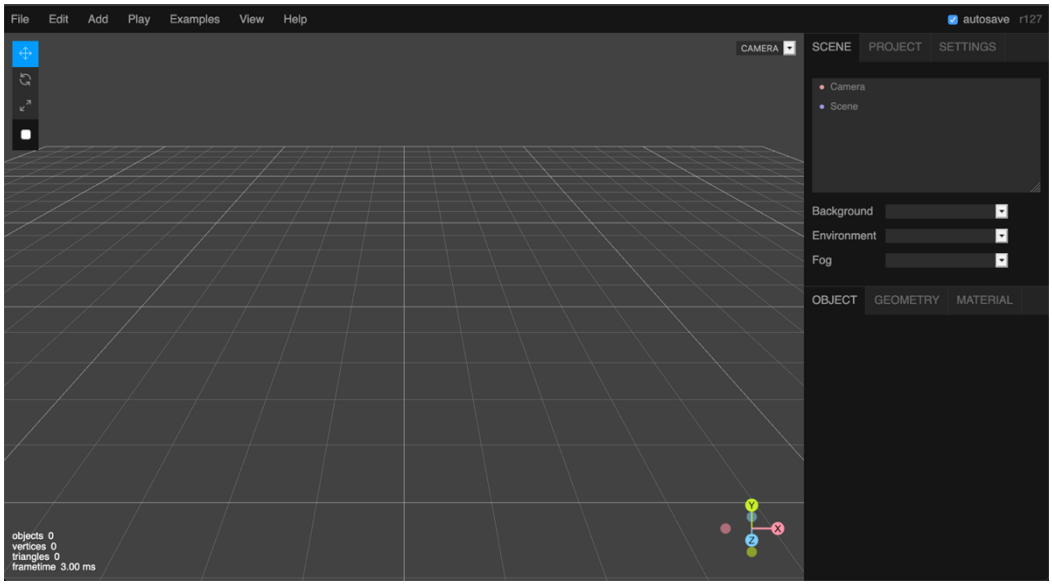

Figura 2.7: Ventana Editor Three.js

El editor de Three.js [12] permite la adición de funcionalidad a los elementos mediante *scripts* escritos en JavaScript, con lo cual serán necesarios conocimientos de este lenguaje para poder usar este editor.

El editor de Three.js [12] es más sencillo que los entornos de desarrollo anteriormente analizados, pero el usuario sigue necesitando de unos conocimientos previos de informática para trabajar con esta herramienta.

2.2.2 Struckd

Struckd [13] es un creador de juegos 3D para Iphone y Android, el cual se ha decidido analizar debido a su sencillez en la interfaz, característica que se quiere ofrecer en el editor de este proyecto, ya que la aplicación que se ha desarrollado está orientada a niños sin experiencia previa en la informática.

Como se puede ver en la Figura 2.8, Struckd [13] consta de una escena del mundo, con una barra inferior que muestra los iconos de los elementos que se pueden añadir a la escena. Struckd [13] utiliza el método Drag & Drop para insertar elementos en la escena.

Struckd [13] es un entorno de desarrollo y un videojuego al mismo tiempo, ya que la idea de esta aplicación es que el usuario pueda crear su propio videojuego con los *assets* facilitados en el editor y posteriormente poder jugar a su propia creación en la misma aplicación.

Struckd [13] ofrece a los usuarios compartir los videojuegos que crean y jugar a los videojuegos creados por otros usuarios de la aplicación.

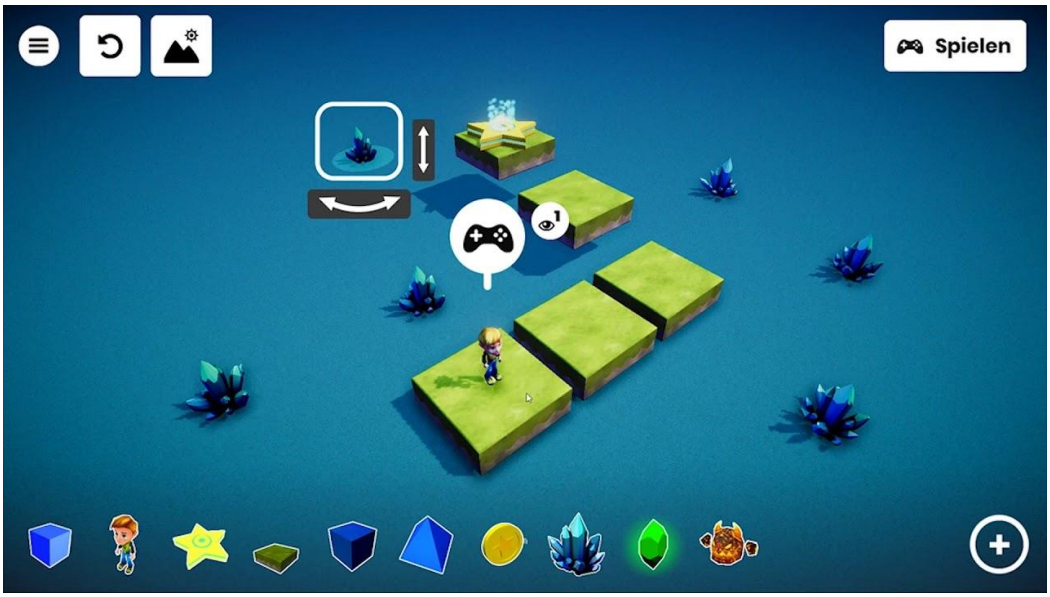

Figura 2.8: Editor Struckd. [14]

3 Diseño

3.1 Análisis

Para realizar un diseño de la aplicación correctamente adaptado a los objetivos principales del proyecto, antes de la creación de la aplicación, se ha realizado un análisis a través de la creación de las tablas de requisitos funcionales y no funcionales, para definir que funciones podrá o no tener la aplicación. Adaptando dichos requisitos a los objetivos del proyecto y así obtener una idea clara de las funciones que la aplicación necesitará poder realizar.

3.1.1 Requisitos funcionales

A través de la siguiente tabla, se definirá de manera clara y sencilla los requisitos funcionales de la aplicación, estos requisitos se pueden ver en la Tabla 3.1.

ID	Requisitos
RF-1	El usuario podrá manejar la cámara del editor mediante el ratón.
RF-2	El editor contará con una opción para la traslación de objetos en la escena.
RF-3	El editor contará con una opción para la rotación de objetos en la escena.
RF-4	El editor contará con una opción para redimensionar los objetos de la escena.
RF-5	El editor contará con una barra con los iconos de los distintos *assets* que se podrán añadir a la escena.
RF-6	Los objetos serán añadidos a la escena del editor mediante el método Drag & Drop, arrastrándolos desde la barra de *assets*.
RF-7	Los *assets* que se mostrarán en la barra de iconos cambiarán según el tipo de *asset* que seleccionemos, a través de un botón en la escena.
RF-8	Mediante el botón izquierdo del ratón, el usuario podrá seleccionar un objeto en la escena.
RF-9	Mediante el botón derecho del ratón, el usuario dejará de tener el objeto seleccionado.
RF-10	La aplicación aplicará los controles de traslación, rotación o cambio de escala (según la opción que este seleccionada mediante un botón en la escena) al objeto que el usuario seleccione.
RF-11	El usuario podrá remover el objeto seleccionado de la escena, presionando el botón suprimir del teclado.

Tabla 3.1: Requisitos funcionales de la aplicación

3.1.2 Requisitos no funcionales

Los requisitos no funcionales de la aplicación quedan definidos en la siguiente tabla, ver Tabla 3.2.

ID	Requisitos
RNF-1	Se utilizará JavaScript, HTML y CSS.
RNF-2	Portabilidad, lograda gracias a los lenguajes utilizados.
RNF-3	Se utilizará la librería de Three.js para realizar el editor y sus funciones.
RNF-4	Interfaz sencilla orientada a niños.

Tabla 3.2: Requisitos no funcionales de la aplicación

3.2 Objetivos

Los objetivos de diseño de la aplicación se pueden dividir en distintos grupos. Desde el punto de vista de la arquitectura de software del proyecto, se quiere lograr una buena escalabilidad de la aplicación para futuras versiones que añadan mejoras y nueva funcionalidad a la aplicación, también se desea que la aplicación sea portable y que se pueda ejecutar en cualquier navegador web. Por otro lado, desde un punto de vista más visual, se quiere lograr una aplicación que tenga una interfaz gráfica sencilla, de fácil uso y orientada a niños y personas sin conocimientos previos de informática.

Para lograr una buena escalabilidad de la aplicación, ha sido un proceso crucial antes de desarrollar el código de la aplicación, plantear una buena modularidad del código, separando la funcionalidad que la aplicación debe tener en distintos grupos conceptualmente. Por tanto, se procedió a realizar un diagrama de clases para dividir la funcionalidad del código en distintos módulos, el cuál describiremos en el siguiente apartado.

La portabilidad se logra gracias al uso exclusivo de los lenguajes JavaScript, HTML y CSS. Utilizando JavaScript para contener toda la lógica de la aplicación y para insertar elementos HTML desde el mismo, y combinando HTML y CSS para el apartado visual de la aplicación.

Por último, la GUI constará de una única ventana que contendrá el editor 3D. No se añadirá información de propiedades de objetos, ni del propio mundo, ya que estos no son los objetivos de la aplicación. En la ventana del editor se añadirán los botones para cambiar los controles sobre el objeto seleccionado, una barra con los *assets* que se podrán añadir al mundo y unos botones que modificarán el contenido de la barra de *assets* con los *assets* correspondientes al botón seleccionado. Se mostrará el resultado de lo aquí mencionado en el capítulo de desarrollo.

3.3 Estructura

A través del diagrama de clases expuesto en este apartado, el cual se puede observar en la Figura 3.1. Se puede apreciar como ha sido distribuida la lógica de la aplicación para conseguir una buena modularidad del código, para que el código quede más organizado y sea más sencillo añadir nueva funcionalidad a través de nuevos módulos o modificar algunos de estos módulos en futuras versiones sin tener que modificar la aplicación al completo.

La aplicación por tanto tiene la siguiente estructura de ficheros:

- Un único fichero HTML: main.htm, el cual contiene en el cuerpo del fichero el *script* que ejecutará el código necesario para ejecutar el editor 3D desarrollado.
- Un único fichero CSS: main.css, el cual contiene todo el estilo de los elementos HTML del editor 3D.
- Una carpeta con todos los *assets* que el editor 3D puede añadir a la escena.
- Una carpeta estructurada en subcarpetas que contiene las imágenes de los iconos que se presentan en la aplicación.
- Seis ficheros JavaScript, que contienen toda la lógica de la aplicación desarrollada:
    - assetsbar.js
    - camera.js
    - editor.js
    - grid.js
    - light.js
    - transformbar.js

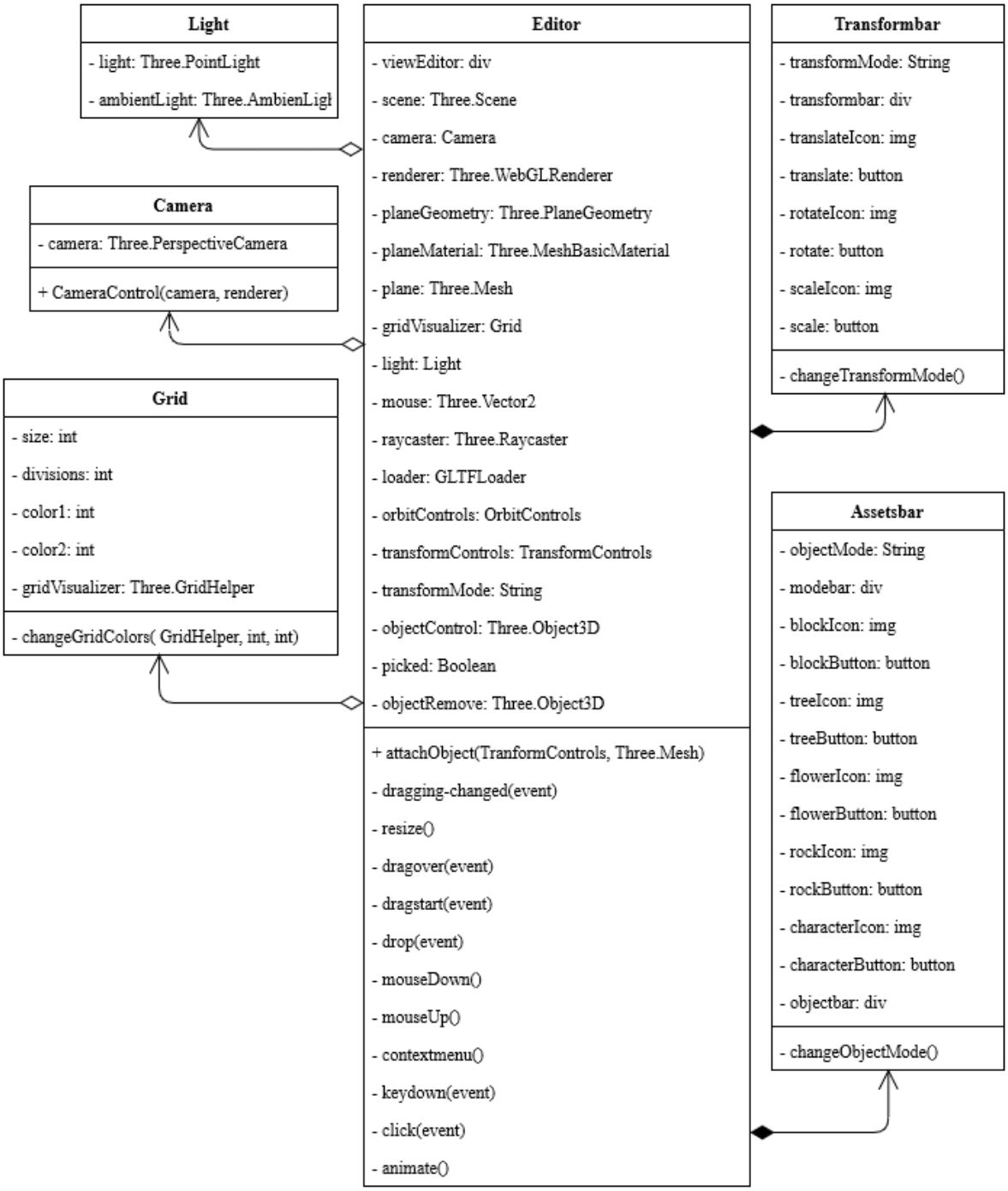

Figura 3.1: Diagrama de clases

4 Desarrollo

Se ha dividido la explicación del desarrollo del proyecto en dos bloques diferenciados, el desarrollo de la aplicación tratando solamente el código realizado, y por otro lado el desarrollo de la parte visual de la aplicación.

En este capítulo se detallará el desarrollo del código de la aplicación, se explicará la funcionalidad que contiene cada fichero realizado y su labor general en la aplicación. Mientras que en el capítulo siguiente del documento, se tratará en detalle el desarrollo visual de la aplicación y se mostrarán los distintos elementos que componen la ventana del editor y el porque se han organizado de dicha manera.

El desarrollo de la aplicación se ha realizado exclusivamente en JavaScript, HTML y CSS. Fue una decisión del proyecto desde el inicio el utilizar la librería Three.js [1] para la realización del editor 3D, es la única librería utilizada para el desarrollo de la aplicación.

Como se ha mencionado con anterioridad en el documento, el código de la aplicación desarrollada consta de: seis ficheros JavaScript, un fichero HTML y un fichero CSS. Estos ficheros contienen toda la funcionalidad de la aplicación. La lógica desarrollada en los ficheros JavaScript se decidió organizarla en seis ficheros, como se ha tratado con anterioridad en el capítulo de diseño.

4.1 HTML

La funcionalidad del fichero main.htm es actuar como el contenedor del editor 3D en la página que se ejecutará en la web. En la cabecera del documento HTML, se encuentra un *link* al fichero de estilos CSS que contiene todas las propiedades de estilo de los elementos HTML. En el cuerpo del fichero HTML, se encuentra un *script* con una llamada a la función Editor que se localiza en el fichero editor.js, el cual iniciará toda la lógica para mostrar el editor 3D en la ventana del navegador.

En el fichero main.htm no se define ningún elemento HTML en la página, todos los elementos HTML han sido definidos desde el propio código de JavaScript en sus correspondientes funciones.

4.2 CSS

Dado que la aplicación solo consta de una ventana y un número no muy elevado de elementos HTML, se ha podido realizar todo el control de estilos de dichos elementos desde un único fichero de manera organizada y de fácil lectura. Esto será de utilidad en el caso de que alguna persona quiera añadir futuras versiones, ya que podrá entender todo lo que este fichero contiene de manera sencilla y en un pequeño rango de tiempo.

4.3 JavaScript

Los ficheros de JavaScript son los que contienen toda la lógica presente para que el editor 3D desarrollado cumpla con todos los requisitos funcionales tratados en el capítulo de diseño. Se va a proceder a explicar que labor realiza el código de cada fichero y el porque se ha realizado de la siguiente manera.

4.3.1 Camera.js

Camera.js contiene la funcionalidad necesaria para crear una cámara con una proyección perspectiva y añadir dicha cámara a la escena del editor en una posición determinada, apuntando al punto central de la escena.

Camera.js también contiene una función para otorgar los controles de movimiento de la cámara al ratón, ofreciendo al usuario el control total de la cámara en la escena del editor 3D.

4.3.2 Grid.js

Grid.js contiene la funcionalidad necesaria para crear una cuadrícula y añadirla a la escena. Aunque en nuestro editor realmente no es necesario realizar ningún tipo de ajuste de simetría a la cuadrícula, se ha decidido crear dicha cuadrícula en la escena ya que otorga al usuario del editor 3D una mejor visión sobre donde empezar a colocar los *assets* en la escena. Con esta pequeña ayuda, la cuadrícula que aparecerá en la ventana del editor, orientará al usuario en los primeros pasos de la creación de la escena 3D.

Grid.js también tiene una función para cambiar los colores de las líneas de la cuadrícula para que tengan un aspecto más agradable y estructurado a primera vista. Se puede observar el resultado de la creación de la cuadrícula en la Figura 4.1.

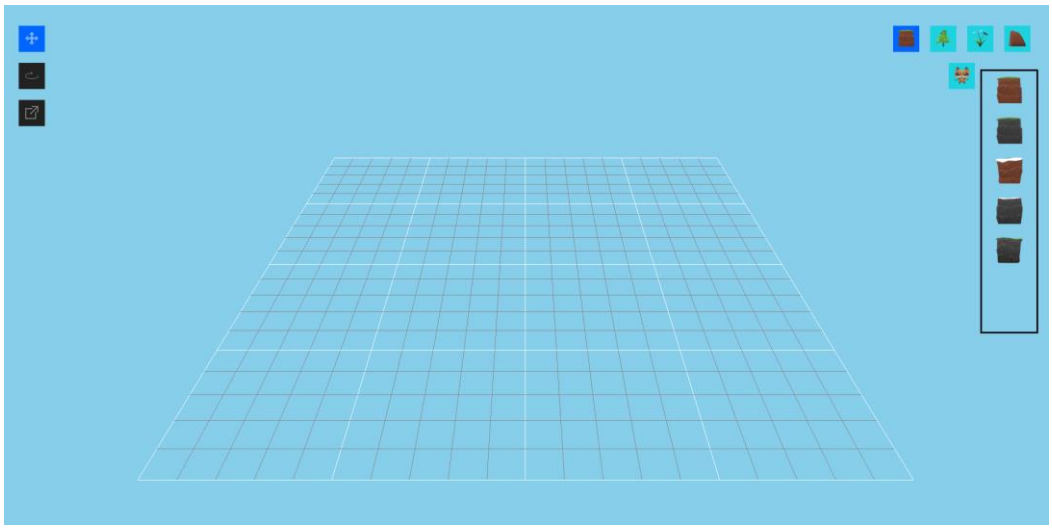

Figura 4.1: Grid, cuadrícula de la escena inicial.

4.3.3 Light.js

Light.js contiene la funcionalidad necesaria para crear una luz ambiente la cuál iluminará toda la escena del editor por igual y añadir dicha luz a la escena del editor 3D.

Se ha decidido añadir la funcionalidad necesaria para crear y añadir a la escena del editor 3D una luz que ilumina desde un punto dado de la escena, para que en futuras versiones con algunas pequeñas modificaciones en el código, el usuario pueda añadir diferentes luces de punto, modificando su color y posición.

4.3.4 Transformbar.js

En Transformbar.js se realiza la creación de una nueva división en el documento HTML, mediante código JavaScript, esta nueva división contiene tres botones con sus correspondientes iconos. El apartado visual de esta creación se tratará en el capítulo de desarrollo visual.

Los tres botones creados se encargan de permitir al usuario elegir el modo de control sobre el objeto que se seleccione. Los tres modos de control posibles son: traslación, rotación y cambio de escala de un objeto. Por tanto, si se selecciona el botón con el modo de rotación, los controles del objeto seleccionado cambiarán a los controles de rotación, o si no se tuviese ningún objeto seleccionado, el siguiente objeto que se seleccione tendrá los controles de rotación. Lo mismo ocurrirá con los modos de control de traslación y cambio de escala.

Por otro lado este fichero también contiene la lógica para cambiar ciertas propiedades HTML de los botones, para que el botón seleccionado se ponga de distinto color gracias a las reglas de estilo definidas en el fichero de CSS.

4.3.5 Assetsbar.js

La funcionalidad desarrollada en Assestsbar.js es muy similar a la lógica desarrollada en el fichero transformbar.js. Aunque en este caso se añade una división que contendrá cinco botones, denominada ModeBar, y otra división que contendrá los iconos de los *assets* que se podrán añadir a la escena mediante el método Drag & Drop, denominada ObjectBar. El apartado visual de estas dos divisiones HTML será tratado en el capítulo de desarrollo visual.

Estos cinco botones, con sus correspondientes iconos añadidos como elementos HTML mediante código JavaScript, se encargan de modificar los iconos que aparecen en la barra de *assets* (creada también en este fichero) en función del botón seleccionado. Es por ello que el fichero Assetsbar.js contiene la funcionalidad necesaria para modificar los iconos mostrados en la barra de *assets* según el botón seleccionado.

El fichero Assetsbar.js también contiene la lógica para mostrar el botón seleccionado de un color distinto al resto, como se ha explicado en el apartado de Transformbar.js.

4.3.6 Editor.js

Editor.js es el fichero más extenso de todos y contiene gran parte de la lógica general para crear el editor 3D. Es por ello que se va a realizar una descripción más detallada del mismo.

Lo primero que realiza este fichero es crear una nueva división HTML que contendrá al editor 3D, y agregar esta división al cuerpo del documento HTML.

En la librería Three.js [1] una escena 3D se compone de una escena, una cámara y un *renderer*. Es por ello que esta es la primera parte de funcionalidad del editor, crear estos tres elementos. Una vez hecho esto, se ajusta el *renderer* al tamaño de la ventana y se añade como hijo a la división HTML del editor 3D.

Posteriormente, se añade un plano a la escena, el cual será la superficie donde se colocarán los distintos objetos mediante el método de Drag & Drop.

Acto seguido, desde Editor.js se llama a la funcionalidad necesaria para crear la cuadrícula y las luces de la escena, y se otorgan los controles de la cámara al ratón mediante la función del fichero camera.js anteriormente mencionada.

Se crea una variable con las coordenadas 2D del ratón y una variable de tipo Three.Raycaster [15] para obtener posteriormente las coordenadas del ratón y el objeto al que se está seleccionando.

A continuación se crean los controles de transformación (traslación, rotación y cambio de escala) y se añaden a la escena.

Ahora se va a definir la funcionalidad de algunos oyentes de eventos y funciones que tiene el módulo editor:

- **draggin-changed:** cuando se activa este evento sobre los controles de transformación, se ejecuta la funcionalidad necesaria para conseguir que no se mueva la cámara de la escena 3D mientras se está utilizando el ratón para realizar cambios en la traslación, rotación o la escala del objeto mediante los controles de transformación.
- **resize:** cuando se activa este evento sobre la ventana del navegador, se ejecuta la funcionalidad necesaria para reajustar el tamaño del *renderer* y modificar la relación de aspecto de la cámara al nuevo tamaño de la página.
- **dragover:** cuando se activa este evento sobre la división HTML del editor 3D, se ejecutan las tareas necesarias para cancelar la funcionalidad por defecto de este evento y dejar así que se puedan soltar elementos en la división del editor exclusivamente.
- **dragstart:** cuando se activa este evento sobre la ventana del navegador, significa que se ha empezado a arrastrar un icono de la barra de *assets* y por ello se realiza la funcionalidad necesaria para obtener el id del objeto que se está arrastrando sobre la ventana.

- **drop:** cuando se activa este evento sobre la división HTML del editor 3D, se realizan las tareas necesarias para que no se ejecute la funcionalidad por defecto de este evento, sino que se aplica una funcionalidad propia. Cuando se suelta el objeto arrastrado sobre la escena del editor 3D, se calcula la intersección de las coordenadas del ratón y el objeto de la escena mediante una recta formada por el punto donde se encuentra el ratón y la posición de la cámara. Una vez obtenida dicha intersección, se obtiene la posición donde se debe cargar el *asset* arrastrado y se crea el *asset* al soltarlo en la escena 3D en dicha posición, y se le otorgan los controles de transformación al objeto. La creación del *asset* como un objeto 3D en la escena es posible gracias a la obtención del id del icono arrastrado, ya que se había guardado el contenido necesario para identificar al objeto que está siendo arrastrado en una variable en el evento dragstart.
- **moueseDown:** cuando se activa este evento sobre los controles de transformación, se otorgan los controles de transformación al objeto que se encuentre en la posición del puntero del ratón.
- **mouseUp:** cuando se activa este evento sobre los controles de transformación, se aplica una funcionalidad necesaria para que el objeto al que se le están aplicando los controles de transformación se mantenga con dichos controles y no cambie a otro objeto que se encuentre en la posición en la que el puntero del ratón termina tras pulsar el botón izquierdo del mismo.
- **contextmenu:** cuando se activa este evento sobre la ventana del navegador, se realiza la funcionalidad necesaria para remover del objeto los controles de transformación. Este evento se activa con el botón derecho del ratón.
- **keydown:** cuando se activa este evento sobre la ventana del navegador, se realiza la funcionalidad necesaria para eliminar el objeto seleccionado de la escena mediante la tecla suprimir.
- **click:** cuando se activa este evento sobre la ventana del navegador, primero se ejecutan las tareas necesarias para que no se realice la funcionalidad por defecto de este evento y una vez hecho esto, se aplica una función propia. Se calculan las coordenadas del ratón y la intersección con los distintos objetos de la escena para otorgar los controles de transformación al primer elemento que se encuentra en la intersección de la recta formada por el punto donde esta el ratón y la posición de la cámara.
- **animate:** es una función que se ejecuta iterativamente, la cual contiene la funcionalidad necesaria para que la escena del editor se recargue cada *frame*.

Por último, se realiza una llamada a la lógica de los ficheros transformbar.js y assetsbar.js para añadir la funcionalidad que tienen estos ficheros al editor 3D.

5 Desarrollo Visual

En este capítulo se va a explicar en detalle como se estructura la aplicación visualmente y que elementos componen la ventana del editor 3D, se puede observar un ejemplo del editor 3D en la Figura 5.1.

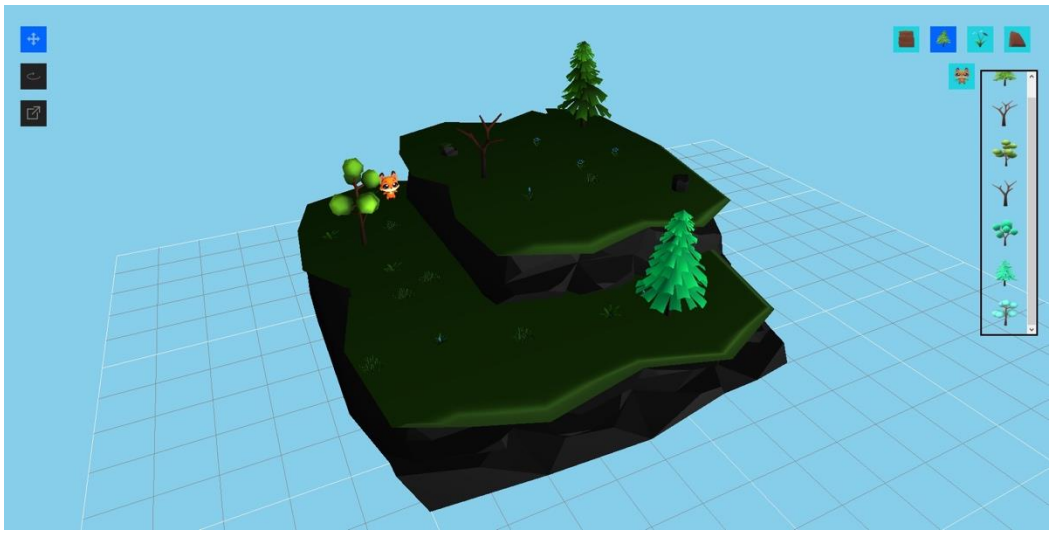

Figura 5.1: Ventana global de mi aplicación

Todos los *assets* utilizados en el editor 3D han sido obtenidos en la página Sketchfab [16], en particular se han usado los conjuntos de assets: PBR Low-poly Fox Character [17], Samar [18], Low poly Stylized Nature Pack [19].

La GUI de la aplicación ha sido diseñada en base al estudio realizado en el capítulo de estudio del arte. El editor tiene una gran influencia tomada de la aplicación Struckd [13] ya que se acerca mucho a los objetivos de la GUI que el proyecto tenía desde un principio, una interfaz sin muchos elementos en pantalla que dificulten su uso, aunque se ha realizado una organización y estructuración de los elementos de la ventana del editor propia. Al mismo tiempo tiene ciertas similitudes con alguna de las características que se han valorado como óptimas de los otros editores analizados.

La GUI de la aplicación se compone de cuatro elementos: ViewEditor, TransformBar, ModeBar y ObjectBar.

Se ha decidido no introducir ningún mensaje de texto en la GUI, de tal forma que toda la interfaz sea visual, propiedad que hace la interfaz más agradable a primera vista y le aporta esa sencillez que se busca como objetivo principal de la aplicación.

5.1 ViewEditor

ViewEditor compone la división del HTML principal ocupando toda la dimensión de la pantalla, se puede observar un ejemplo en la Figura 5.2. Esta división la compone el *renderer* del editor 3D, por lo que la escena del editor 3D se encuentra en el elemento HTML ViewEditor. ViewEditor tiene la propiedad necesaria para que se puedan soltar los iconos del ObjectBar sobre la misma, mediante el método Drag & Drop y que se ejecute el código necesario para cargar dicho objeto en la escena en la posición del ratón.

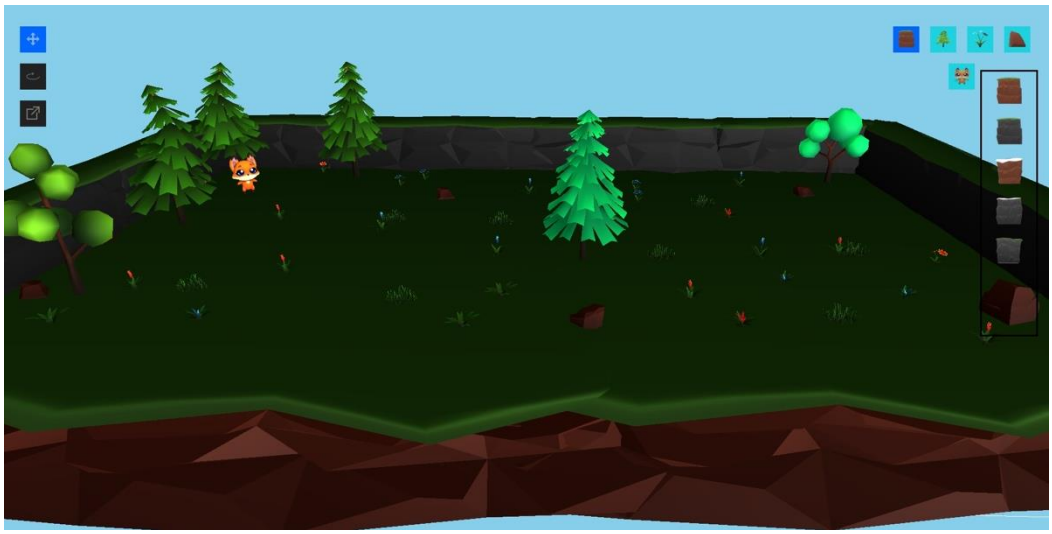

Figura 5.2: ViewEditor

5.2 TransformBar

TransformBar es una división de HTML, se encuentra en la parte superior izquierda a modo de columna, podemos observar su apariencia en la Figura 5.3. TransformBar está compuesta por tres botones con sus correspondientes iconos añadidos como imágenes a dichos botones. Estos botones una vez presionados hacen una llamada a la funcionalidad descrita en el apartado de desarrollo de código. El primer botón activa el modo de traslación, el segundo activa el modo de rotación y el tercero activa el modo de cambio de escala.

El botón que está seleccionado actualmente presenta un color de fondo en azul oscuro para diferenciarlo del resto de botones no seleccionados.

Figura 5.3: TransformBar

5.3 ModeBar

ModeBar es una división de HTML, se encuentra en la parte superior derecha a modo de columna horizontal, se puede observar su aparienca en la Figura 5.4. ModeBar está formado por dos filas de cinco botones en total, cada botón modifica los iconos de los *assets* que se muestran en ObjectBar.

Al igual que en TransformBar como se ha explicado anteriormente, el botón seleccionado se muestra con un color de fondo en azul oscuro.

Figura 5.4: ModeBar

5.4 ObjectBar

ObjectBar es una división de HTML localizada en el lateral derecho de la pantalla junto con ModeBar, se pueden observar los distintos iconos que se muestran en ObjectBar dependiendo del botón pulsado de ModeBar en la Figura 5.5. En ObjectBar se encuentran los iconos de los *assets* que se pueden introducir en la escena de ViewEditor mediante el método de Drag & Drop.

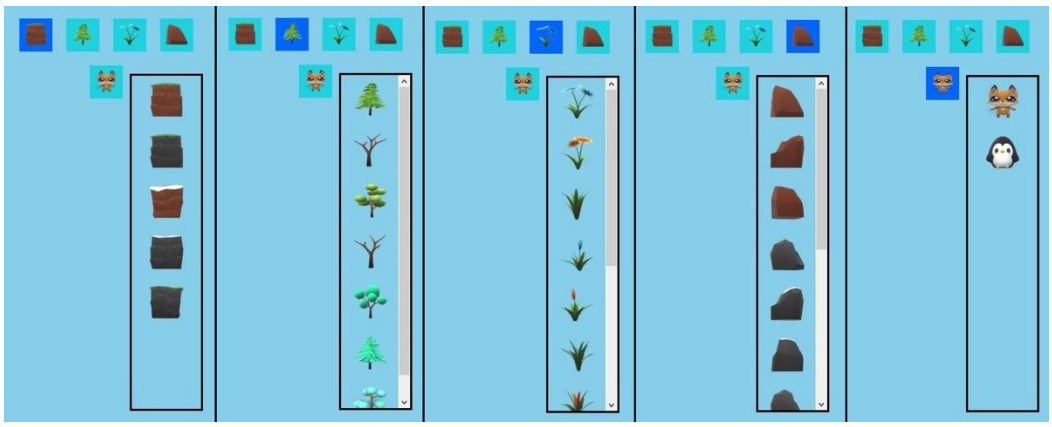

Figura 5.5: ObjectBar, con las cinco posibles opciones de ModeBar.

6 Integración, pruebas y resultados

En este capítulo se abordará como se ha llevado a cabo la realización de las pruebas cada vez que se añadía nueva funcionalidad al editor 3D, la cual se debía de verificar que funcionaba correctamente para integrarla definitivamente en el código de la aplicación.

Cabe mencionar en este capítulo, que la aplicación se ha ejecutado en un servidor local, mediante una extensión de Visual Studio Code llamada Live Server publicada por Ritwick Dey [20]. La aplicación puede ser ejecutada con otros métodos, a través de un servidor local, si el usuario así lo desea.

Dado que la aplicación funciona en una ventana del navegador, la primera tarea una vez desarrollada la nueva funcionalidad y verificar que no existía ningún error léxico o algún bloque mal estructurado, era ejecutar la aplicación localmente en el navegador.

Una vez ejecutada la aplicación en el navegador, la siguiente tarea era verificar que no había ninguna advertencia o error en la consola del navegador independientemente del tipo de nueva funcionalidad que se estuviese probando. Para verificar esto, se debía presionar la tecla F12 para obtener en la pantalla del navegador información para desarrolladores.

En los primeros estados de desarrollo de la aplicación se obtuvieron algunas advertencias sobre la librería Three.js [1], debido a que en distintos ficheros se referenciaba a la librería desde distintas ubicaciones y se creaban distintos espacios para esta librería, aunque esto no provocaba el mal funcionamiento de la aplicación, era una mala práctica y por ello fue importante verificar fichero a fichero que la procedencia de la librería de Three.js [1] fuese la misma en cada uno de ellos.

Cuando se añadía nueva funcionalidad a la aplicación, en algunas ocasiones se tenían pérdidas en el contenido de algunas variables. Para solucionar este problema se tuvo que revisar la estructuración del código y modificar algunas llamadas a funciones. Estos errores ocurrían debido a que hay partes del código que se actualizan constantemente ya que se refresca la pantalla cada *frame*, por tanto se tuvo que tener dicha situación en cuenta y estructurar el código correctamente.

Una vez ejecutada la aplicación con la nueva funcionalidad a probar y verificado que no hay presentes ninguna advertencia ni error en la consola, el último paso era probar en el editor 3D que la funcionalidad desarrollada funcionaba correctamente mediante la realización de escenas 3D de prueba que pusiesen dicha nueva funcionalidad en práctica.

Por otro lado, cuando se ha probado un nuevo módulo el cual introducía elementos HTML mediante código JavaScript, se utilizaba el inspector de estilos y DOM (herramienta ofrecida en la ventana al presionar la tecla F12 para obtener información para desarrolladores en la ventana del navegador) para verificar la correcta creación de los elementos HTML y su correcta posición en la ventana del navegador. También se verificaba que los elementos tenían las propiedades de estilo definidas en el archivo CSS bien aplicadas sobre cada elemento HTML.

Por tanto, para cada prueba de una nueva funcionalidad ha sido necesario comprobar el correcto funcionamiento de dicha funcionalidad manualmente, adquiriendo el papel de usuario de la propia aplicación y de esta manera verificar que se realizaban de forma correcta todas las posibles acciones que el usuario pudiese realizar con la nueva funcionalidad desarrollada.

Una vez se realizaban todas las pruebas para verificar el correcto funcionamiento de la nueva funcionalidad, se estructuraba correctamente el código, creando un nuevo módulo si se viese necesario o simplemente se realizaba una buena estructuración del mismo en el fichero Editor.js. Siempre que fuese necesario, se ha realizado una pequeña documentación en el propio fichero de código para documentar la nueva funcionalidad, esto ayudará en trabajos futuros a entender y comprender rápidamente el código de nuevo.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Para obtener una conclusión final sobre el proyecto que se ha realizado, se va a analizar como se ajusta el trabajo final a los objetivos iniciales del proyecto.

Es primordial valorar como se ajusta la aplicación final al objetivo principal del proyecto, este es, realizar un editor 3D orientando a niños sin conocimientos previos de informática para el aprendizaje de conocimientos básicos en la creación de una escena 3D, para iniciarse en el desarrollo de videojuegos.

Como se ha explicado con anterioridad en este documento, muchas de las decisiones tomadas durante el desarrollo y el diseño de la aplicación han sido con el objetivo principal de la aplicación siempre en mente. Gracias a esta buena práctica y estructuración del desarrollo de la aplicación se puede concluir que se ha realizado una buena labor en este aspecto. Se ha realizado una GUI bastante amigable para el usuario, fácil de entender y usar sin la necesidad de conocimientos previos de informática.

Con la aplicación web actual hay presentes ciertas limitaciones que serán abordadas en un trabajo futuro y se hablarán sobre las mismas en el siguiente apartado. Pese a esto, el editor 3D desarrollado cumple con los objetivos principales del proyecto y con los requisitos funcionales que se pretendían conseguir.

Se ha desarrollado una aplicación que funciona en la web, mediante el uso de los lenguajes: JavaScript, HTML y CSS. Con una GUI sencilla y fácil de entender para el usuario, y un número limitado de *assets* que se pueden añadir a la escena mediante el uso del método de Drag & Drop. Se pueden conseguir realizar escenas 3D que bien podrían comprender el mundo o nivel de un videojuego 3D gracias al control de estos objetos 3D en la escena, mediante traslación, rotación y cambio de escala de dichos objetos.

7.2 Trabajo futuro

Ya que el proyecto que se ha desarrollado es un editor 3D, son muchas las nuevas funcionalidades que podría presentar la aplicación para hacerla más completa. Siempre respetando los objetivos principales de la aplicación. Por tanto, en esta sección se van a describir los nuevos añadidos que según he valorado individualmente, pienso que aportarán un gran progreso de funcionalidad a la aplicación.

- **Guardar la escena 3D:** es evidente pensar que este es uno de los trabajos futuros más importantes a realizar, ya que ahora se ha conseguido realizar los objetivos del proyecto y desarrollar una aplicación capaz de crear escenas 3D para videojuegos (u otros propósitos) orientada a niños. El siguiente paso es crear la funcionalidad para que el editor pueda guardar la escena 3D en un archivo y cargar dicha escena en cualquier otro momento en el editor 3D desarrollado en este proyecto o en otro editor 3D que no sea la propia aplicación.
- **Controles en primera o tercera persona para los personajes:** dado que el editor está orientado a la creación de escenas 3D para el desarrollo de videojuegos, aunque estas escenas 3D bien podrían ser realizadas con otros propósitos. Añadir controles a los personajes de la escena permitiría a mi editor tener la capacidad de realizar pruebas en vivo de como sería jugar en dicha escena 3D creada por el usuario.
- **Añadir físicas entre objetos:** esta característica servirá para seguir completando la funcionalidad total de la aplicación, se realizará la funcionalidad necesaria para la interacción física entre objetos.

Referencias

[1] Three.js, [Online]. Available at: https://threejs.org/
[Accedido febrero 25, 2021]

[2] Roblox, [Online]. Available at: https://www.roblox.com/
[Accedido abril 05, 2021]

[3] Fortnite, [Online]. Available at: https://www.epicgames.com/fortnite/es-ES/home
[Accedido abril 05, 2021]

[4] Unity, [Online]. Available at: https://unity.com/es
[Accedido marzo 25, 2021]

[5] CryEngine, [Online]. Available at: https://www.cryengine.com/
[Accedido marzo 25, 2021]

[6] Unreal Engine, [Online]. Available at: https://www.unrealengine.com/en-US/
[Accedido marzo 25, 2021]

[7] *Extraído de:* https://answers.unity.com/storage/attachments/164669-camera-macro.png

[8] *Extraído de:* https://cdn.80.lv/api/upload/content/78/5d2749d2a5acc.jpg

[9] *Extraído de:* https://docs.cryengine.com/download/attachments/26877242/flowMain.png?version=1&modificationDate=1519886016000&api=v2

[10] *Extraído de:* https://i.imgur.com/XNmXLUg.jpg

[11] *Extraído de:* https://4.bp.blogspot.com/-FlX2LOzT1_8/WRAoQlxa3TI/AAAAAAAAAnc/EnOyIUO8n9sz9LO0ZMXSkuydrXQbnZq8QCLcB/s1600/unreal-engine-4-blueprint-editor.jpg

[12] Three.js - Editor, [Online]. Available at: https://threejs.org/editor/
[Accedido febrero 26, 2021]

[13] Struckd - 3D Game Creator, [Online]. Available at: https://struckd.com/
[Accedido abril 05, 2021]

[14] *Extraído de:* https://i.ytimg.com/vi/b-lAju0RDJI/maxresdefault.jpg

[15] Three.js - Raycaster, [Online]. Available at:
https://threejs.org/docs/index.html?q=raycas#api/en/core/Raycaster
[Accedido marzo 29, 2021]

[16] Sketchfab, [Online]. Available at: https://sketchfab.com/
[Accedido abril 07, 2021]

[17] "PBR Low-poly Fox Character" (https://skfb.ly/6SpGT) by Ida Faber is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[18] "Samar" (https://skfb.ly/6VGCQ) by Astra is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[19] "Low poly Stylized Nature Pack" (https://skfb.ly/6VZTt) by Satendra Saraswat is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[20] Live Server, Ritwick Dey. [Online]. Available at: https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
[Accedido febrero 26, 2021]

Anexos

A Manual de instalación

Como se ha explicado en el subcapítulo 3.3 de este documento, la aplicación tiene la siguiente estructura de ficheros:

- Un único fichero HTML: main.htm, el cual contiene en el cuerpo del fichero el script que ejecutará el código necesario para ejecutar el editor 3D desarrollado.
- Un único fichero CSS: main.css, el cual contiene todo el estilo de los elementos HTML del editor 3D.
- Una carpeta con todos los *assets* que el editor 3D puede añadir a la escena.
- Una carpeta estructurada en subcarpetas que contiene las imágenes de los iconos que se presentan en la aplicación.
- Seis ficheros JavaScript, que contienen toda la lógica de la aplicación desarrollada:
    - assetsbar.js
    - camera.js
    - editor.js
    - grid.js
    - light.js
    - transformbar.js

Para ejecutar el editor 3D en el navegador se debe ejecutar un servidor local, para realizar esta acción hay múltiples posibilidades, como se puede observar en la documentación de la librería Three.js [1] utilizada. Aunque en este manual se va a explicar la opción que ha sido utilizada durante el desarrollo y las pruebas del proyecto.

Se debe abrir el proyecto con Visual Studio Code e instalar la extensión: Live Server, publicada por Ritwick Dey [20]. Una vez se ha instalado esta extensión, se debe hacer *click* con el botón derecho del ratón, sobre el fichero main.htm en el explorador de archivos del proyecto que ofrece Visual Studio Code y elegir la opción: *Open with* Live Server (Abrir con Live Server). A continuación, la aplicación se abrirá en una ventana del navegador.

Glosario

JS	JavaScript
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
GUI	Graphical User Interface
DOM	Document Object Model